Q1:

Person.h

#ifndef Q1\_PERSON\_H  
#define Q1\_PERSON\_H  
  
#include <string>  
  
using namespace std;  
  
class Person  
{  
private:  
 string firstName;  
 string middleName;  
 string lastName;  
 string street;  
 string city;  
 string country;  
  
  
public:  
 // Constructors #1  
 Person();  
 // Constructors #2  
 Person(string firstName, string middleName, string lastName, string street, string city, string country);  
  
 // Mutator functions  
 void setfirstName(string firstName);  
 void setmiddleName(string middleName);  
 void setlastName(string lastName);  
 void setStreet(string street);  
 void setCity(string city);  
 void setCountry(string country);  
  
  
 // Accessing functions  
 string getfirstName() const;  
 string getmiddleName() const;  
 string getlastName() const;  
 string getStreet() const;  
 string getCity() const;  
 string getCountry() const;  
};  
  
#endif //Q1\_PERSON\_H

Person.cpp

#include "Person.h"  
  
// Constructor #1  
Person::Person()  
{  
 firstName = "";  
 middleName = "";  
 lastName = "";  
 street = "";  
 city = "";  
 country = "";  
}  
  
// Constructor #2  
Person::Person(string firstName, string middleName, string lastName, string street, string city, string country)  
{  
 this->firstName = firstName;  
 this->middleName = middleName;  
 this->lastName = lastName;  
 this->street = street;  
 this->city = city;  
 this->country = country;  
}  
  
  
void Person::setfirstName(string firstName)  
{  
 this->firstName = firstName;  
}  
  
void Person::setmiddleName(string middleName)  
{  
 this->middleName = middleName;  
}  
  
void Person::setlastName(string lastName) {  
 this->lastName = lastName;  
}  
  
void Person::setStreet(string street) {  
 this->street = street;  
}  
  
void Person::setCity(string city) {  
 this->city = city;  
}  
  
void Person::setCountry(string country) {  
 this->country = country;  
}  
  
// Accessor functions  
string Person::getfirstName() const {  
 return firstName;  
}  
  
string Person::getmiddleName() const {  
 return middleName;  
}  
  
string Person::getlastName() const {  
 return lastName;  
}  
  
string Person::getStreet() const {  
 return street;  
}  
  
string Person::getCity() const  
{  
 return city;  
}  
  
string Person::getCountry() const {  
 return country;  
}

Address.h

#ifndef Q1\_ADDRESS\_H  
#define Q1\_ADDRESS\_H  
  
#include "Person.h"  
  
  
// Address class declaration  
class Address : public Person  
{  
private:  
 int block;  
 int unit;  
 int floor;  
 int postalCode;  
  
  
public:  
 // Constructors #1  
 Address();  
 // Constructors #2  
 Address(string firstName, string middleName, string lastName, string street, string city, string country, int block, int unit, int floor, int postalCode);  
  
 // Mutator functions  
 void setBlock(int block);  
 void setUnit(int unit);  
 void setFloor(int floor);  
 void setPostalCode(int postalCode);  
  
 // Accessing functions  
 int getBlock() const;  
 int getUnit() const;  
 int getFloor() const;  
 int getPostalCode() const;  
  
};  
  
  
#endif //Q1\_ADDRESS\_H

Address.cpp

#include "address.h"  
  
// Constructor #1  
Address::Address()  
{  
 block = 0;  
 unit = 0;  
 floor = 0;  
 postalCode = 0;  
}  
  
// Constructor #2  
Address::Address(string firstName, string middleName, string lastName, string street, string city, string country, int block, int unit, int floor, int postalCode)  
{  
 // Set attributes derived from Person  
 setfirstName(firstName);  
 setmiddleName(middleName);  
 setlastName(lastName);  
 setStreet(street);  
 setCity(city);  
 setCountry(country);  
  
  
 // Set Address attributes  
 this->block = block;  
 this->unit = unit;  
 this->floor = floor;  
 this->postalCode = postalCode;  
}  
  
// Mutator functions  
void Address::setBlock(int block) {  
 this->block = block;  
}  
  
  
void Address::setUnit(int unit) {  
 this->unit = unit;  
}  
  
void Address::setFloor(int floor) {  
 this->floor = floor;  
}  
  
void Address::setPostalCode(int postalCode) {  
 this->postalCode = postalCode;  
}  
  
// Accessor functions  
int Address::getBlock() const {  
 return block;  
}  
  
int Address::getUnit() const {  
 return unit;  
}  
  
int Address::getFloor() const {  
 return floor;  
}  
  
int Address::getPostalCode() const {  
 return postalCode;  
}

Main.cpp

#include <iostream>  
#include "Address.h"  
  
// Function to display the information of person  
void DisplayPerson(Address c)  
{  
 string result;  
  
 // Display all information of the person  
 cout << "-----------" << endl  
 << "First Name: " << c.getfirstName() << endl  
 << "Middle Name: " << c.getmiddleName() << endl  
 << "Last Name: " << c.getlastName() << endl  
 << "Street: " << c.getStreet() << endl  
 << "City: " << c.getCity() << endl  
 << "Country: " << c.getCountry() << endl  
 << "Block: " << c.getBlock() << endl  
 << "Unit: " << c.getUnit() << endl  
 << "Floor: " << c.getFloor() << endl  
 << "postalCode: " << c.getPostalCode() << endl;  
}  
  
int main()  
{  
 // Create the first instance of person  
 Address person1;  
  
 // Add information to person1  
 person1.setfirstName("Joan");  
 person1.setmiddleName("Smith");  
 person1.setlastName("Hank");  
 person1.setStreet("123 Main Street");  
 person1.setCity("Seattle");  
 person1.setCountry("the US");  
 person1.setBlock(45);  
 person1.setUnit(110);  
 person1.setFloor(2);  
 person1.setPostalCode(12345);  
  
 // Display the information of person 1  
 cout << "Person #1" << endl;  
 DisplayPerson(person1); // Call function DisplayPerson  
  
 // Create the second instance of Person and add information to it  
 Address person2("Jones", "Jenny", "Wang", "555 East Street", "Portland", "the US", 88, 36, 5, 654321);  
  
 // Display the information of person 2  
 cout << "Person #2" << endl;  
 DisplayPerson(person2); // Call function DisplayPerson  
  
 return 0;  
}

Person #1

-----------

First Name: Joan

Middle Name: Smith

Last Name: Hank

Street: 123 Main Street

City: Seattle

Country: the US

Block: 45

Unit: 110

Floor: 2

postalCode: 12345

Person #2

-----------

First Name: Jones

Middle Name: Jenny

Last Name: Wang

Street: 555 East Street

City: Portland

Country: the US

Block: 88

Unit: 36

Floor: 5

postalCode: 654321

Q2:

Set\_Union

#include <algorithm> // std::set\_union, std::sort

#include <iostream> // std::cout

#include <vector> // std::vector

// Driver code

int main()

{

int first[] = { 4, 8, 12, 16, 20 };

int second[] = { 40, 36, 32, 28, 24 };

int n = sizeof(first) / sizeof(first[0]);

// Print first array

std::cout << "First array contains :";

for (int i = 0; i < n; i++)

std::cout << " " << first[i];

std::cout << "\n";

// Print second array

std::cout << "Second array contains :";

for (int i = 0; i < n; i++)

std::cout << " " << second[i];

std::cout << "\n\n";

std::vector<int> v(10);

std::vector<int>::iterator it, st;

std::sort(first, first + n);

std::sort(second, second + n);

// Using default function

it = std::set\_union(first, first + n, second,

second + n, v.begin());

std::cout << "The union has " << (it - v.begin())

<< " elements:\n";

for (st = v.begin(); st != it; ++st)

std::cout << ' ' << \*st;

std::cout << '\n';

return 0;

}

-----

First array contains : 4 8 12 16 20

Second array contains : 40 36 32 28 24

The union has 10 elements:

4 8 12 16 20 24 28 32 36 40

Set\_Intersection

#include <iostream>

#include <algorithm>

#include <vector>

bool comp(int a, int b)

{

return a < b;

}

int main()

{

int first[] = { 16, 20, 24, 28, 32 };

int second[] = { 40, 36, 32, 28, 24 };

int n = sizeof(first) / sizeof(first[0]);

std::vector<int> v1(5);

std::vector<int> v2(5);

std::vector<int>::iterator it, ls;

std::sort(first, first + 5);

std::sort(second, second + 5);

// Print elements

std::cout << "First array :";

for (int i = 0; i < n; i++)

std::cout << " " << first[i];

std::cout << "\n";

// Print elements

std::cout << "Second array :";

for (int i = 0; i < n; i++)

std::cout << " " << second[i];

std::cout << "\n\n";

// std :: set\_intersection

ls = std::set\_intersection(first, first + 5, second, second + 5, v1.begin(), comp);

std::cout << "The intersection has " << (ls - v1.begin()) << " elements:";

for (it = v1.begin(); it != ls; ++it)

std::cout << ' ' << \*it;

std::cout << "\n";

return 0;

}

----

First array : 16 20 24 28 32

Second array : 24 28 32 36 40

The intersection has 3 elements: 24 28 32

Set\_Subset

#include <iostream>

using namespace std;

void combinationUtil(int arr[], int n, int r,

int index, int data[], int i);

void printCombination(int arr[], int n, int r)

{

int data[r];

combinationUtil(arr, n, r, 0, data, 0);

}

/\* arr[] ---> Input Array

n ---> Size of input array

r ---> Size of a combination to be printed

index ---> Current index in data[]

data[] ---> Temporary array to store current combination

i ---> index of current element in arr[] \*/

void combinationUtil(int arr[], int n, int r, int index,

int data[], int i)

{

// Current combination is ready, print it

if (index == r) {

for (int j = 0; j < r; j++)

cout <<" "<< data[j];

cout <<"\n";

return;

}

// When no more elements are there to put in data[]

if (i >= n)

return;

// current is included, put next at next location

data[index] = arr[i];

combinationUtil(arr, n, r, index + 1, data, i + 1);

// current is excluded, replace it with next

// (Note that i+1 is passed, but index is not

// changed)

combinationUtil(arr, n, r, index, data, i + 1);

}

// Driver program to test above functions

int main()

{

int arr[] = { 20, 24, 28, 32, 36 };

int r = 3;

int n = sizeof(arr) / sizeof(arr[0]);

printCombination(arr, n, r);

return 0;

}

----

20 24 28

20 24 32

20 24 36

20 28 32

20 28 36

20 32 36

24 28 32

24 28 36

24 32 36

28 32 36